**Microservice WMS Billing Module**

1. **INTRODUCTION**

The intention of the document would be providing the scope, plan and milestone for our application. I have segregated the idea behind the application in two mode as “Basic” and “Advanced” and both facilitate the same functionality but differs mainly on platform agility.

We start with “Basic” and migrate the same to “advanced” mode if time permits.

1. **APPLICATION STACK**

* Java Spring-boot as the microservice platform layer.
* Cassandra as our data layer.
* Docker based application.

Note: With assumption that the local system has Docker installed in it,

* <https://hub.docker.com/r/bitnami/cassandra/> - Link to get and run the Cassandra in one node-mode and would appreciate the team to get the hands-on with simple queries. First timer please follows the steps provided under title “Connecting to Other containers” -> “Using the command line”
* <https://github.com/datastax/java-driver> - Link to get the Java Database connectivity with Cassandra library.

1. **SCOPE and COMPONENT ROLE**

The intension is to replace the Sterling WMS out-of-the-box ARE module with NoSql based microservice.

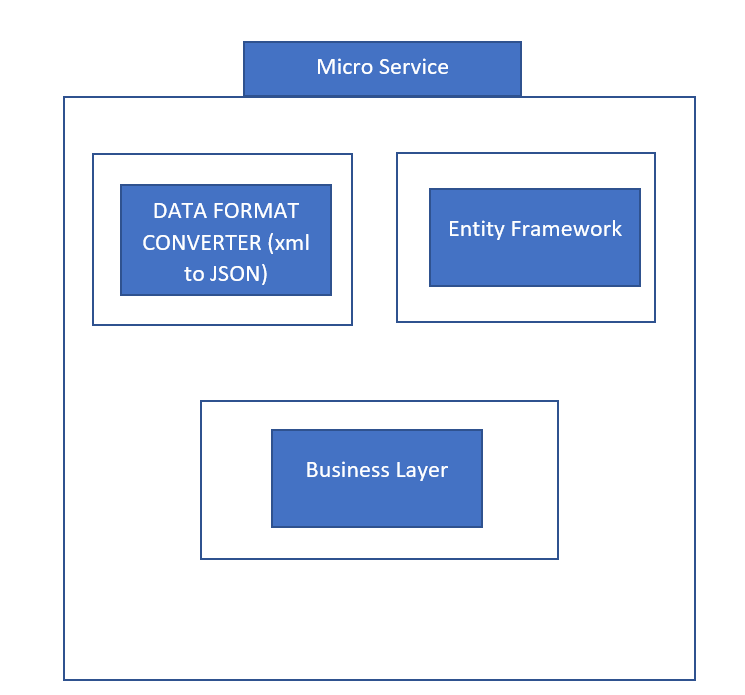
**3.1. MICROSERVICE**

The microservice responsible to process the request from STERLING (JSON based input POST request) and it should have its own Entity Framework to communicate with Cassandra Database. The POST request must be one of the following types,

* INSERT POST request to insert the Activity detail/Activity Header into Cassandra.
* Update POST request to change the Activity detail/Activity Header information in Cassandra.
* DELETE POST request to change the Activity detail/Activity Header information in Cassandra.
* GET POST request to filter the data records from Cassandra

**To be implement:**

* + Entity Framework to process DB request such as INSERT, SELECT, UPDATE and so forth.
  + Business Layer to process the Activity based on the transaction type, transaction data and activity code such as receive a line, receive a shipment, pack a shipment and so forth.

**Logical Design:** 

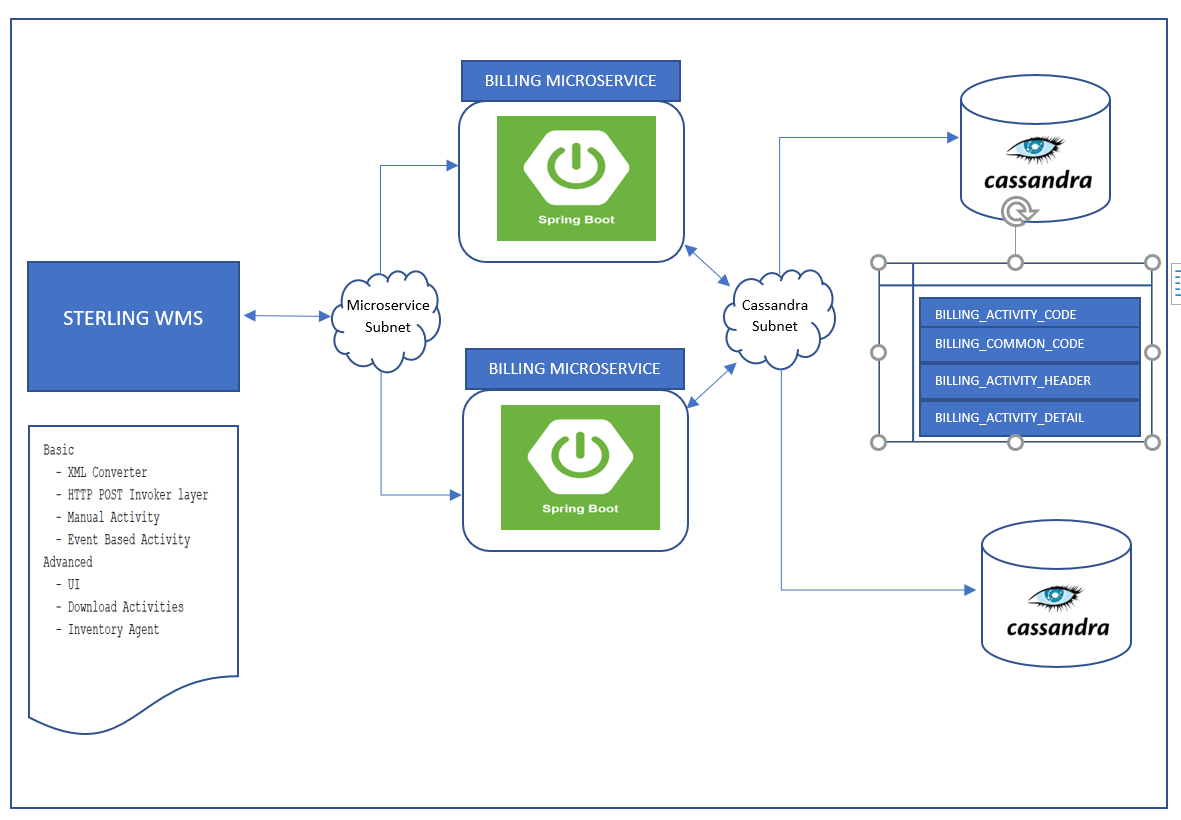
**3.2.**  **STERLING CUSTOMIZATION:**

All the ARE related data should be retained ONLY in Cassandra and accessed via microservice. We should implement the REST POST service layer for STERLING SDF service to communicate with microservice on synchronous mode.

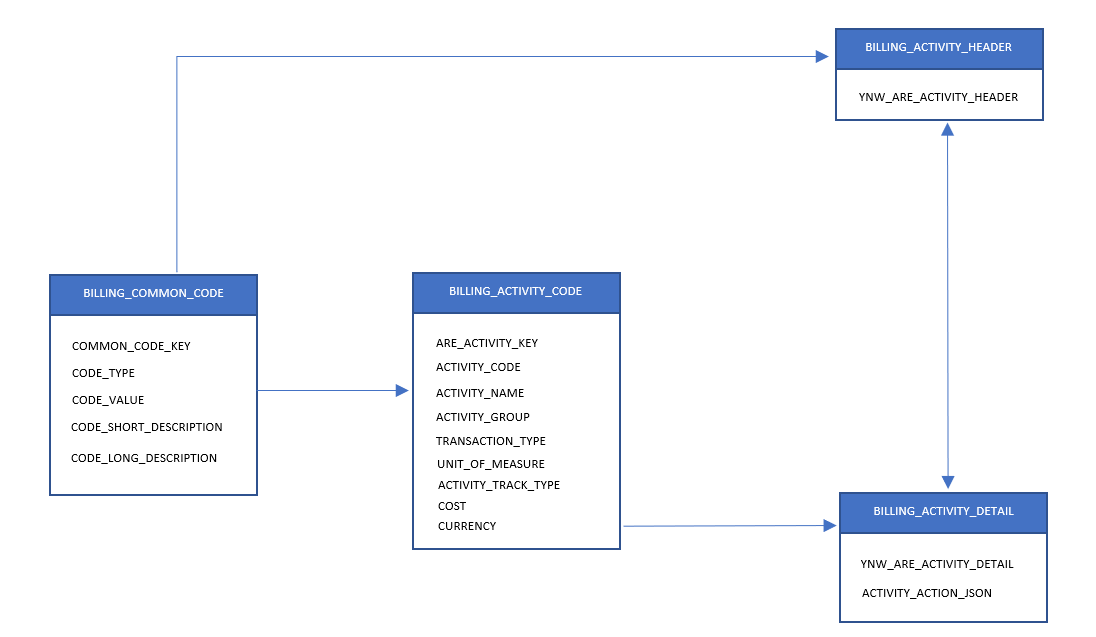
Transactions such as receiving, Putaway, picking and so forth must be enable with associated transaction events and triggering the same would invoke SDF sterling services which internally calls microservice to record the activities in Cassandra.

The ARE related Out-of-the-box console should be customized (Replace the Sterling ARE related API’s with SDF services which calls microservice with relevant PUT, POST or GET request)

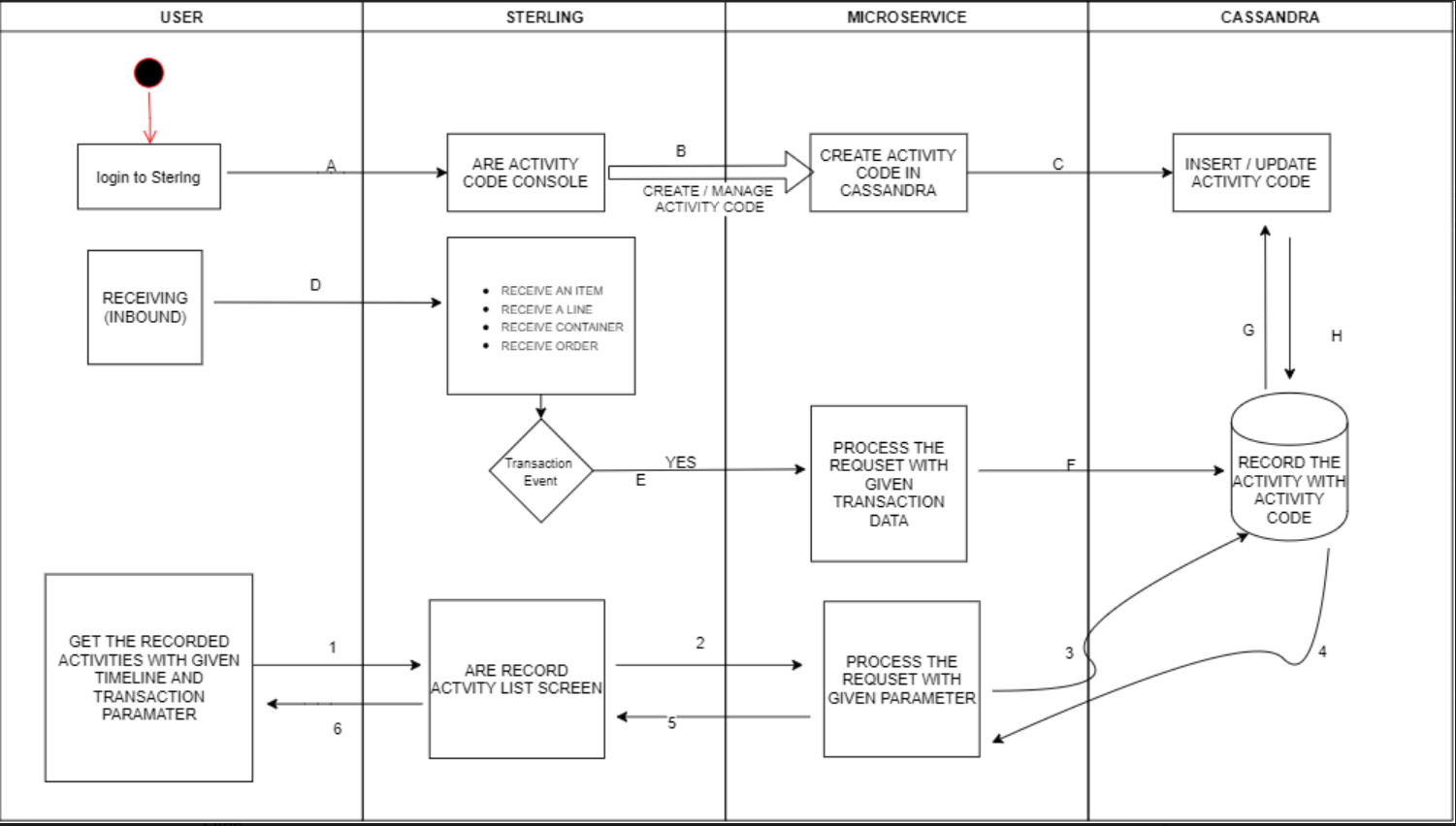
1. **ARCHITECTURAL LAYOUT FOR BILLING MODULE**



1. **ENTITY DATA MODEL**



1. **APPLICATION DATA FLOW**



1. **MILESTONE**

**Phase I:**  The components mentioned on this phase could be implemented by parallel mode as in one may not affect the others. So, team can be segregate into three and to achieve the components.

* Entity Framework for Casandra tables. Utilize <https://github.com/datastax/java-driver> resource for Cassandra DB connection library. i.e) DML should be happens through entity objects in microservice Springboot code.
* Webservice generic communication platform in Sterling instance. i.e:) Should implement generic method/function in Sterling to establish handshake and channel to Springboot microservice which later be used by Sterling SDF Services to make a Synchronous call with microservice.
* Springboot microservice implementation to handle the JSON POST Request.

Note: At end this Phase I, team should have the working prototype of

* Entity Framework (We could be Insert, Update, select records in Cassandra using Object Via Entity Framework)
* Using POSTMAN, we could be interacting with Springboot microservice with JSON POST method.
* From Sterling SDF Service, we could be able channel with Springboot Service.

**Phase II:**

* Integrate the Entity Framework and Springboot microservice
* Implement the ARE Activity logics in Springboot microservice. Eg: Bind the Activity code for Activities completed in Sterling WMS and record the same in Cassandra using the Entity Framework.

Note: At end of this Phase II,

* Team can be performing the ARE Activities recording through microservice using POSTMAN by mocking the Input JSON (Assuming the same would be coming from Sterling WMS going forward)

**Phase III:**

* Create the SDF Services for the Activities we would like to capture such as receiving an order line, receiving complete Order ad so forth
* Integrate the same with provided generic webservice methods. (If it works through POSTMAN on Phase II, then it should work also through Sterling SDF Services.)